Entity:  Database entity is **a thing, person, place, unit, object or any item about which the data should be captured and stored in** the form of properties, workflow and tables.

Entity relationship diagrams:

* ER model allows us to sketch database designs
* ERD is a graphical tool for modeling data.
* ERD is widely used in database design
* ERD is a graphical representation of the logical structure of a database
* ERD is a model that identifies the concepts or entities that exist in a system and the relationships between those entities
* Purposes:
  + The database analyst/designer gains a better understanding of the information to be contained in the database through the process of constructing the ERD.
  + The ERD serves as a documentation tool.
  + Finally, the ERD is used to communicate the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.
* Classification of a relationship
  + Optional Relationship
    - An Employee may or may not be assigned to a Department
    - A Patient may or may not be assigned to a Room
  + Mandatory Relationship
    - Every Course must be taught by at least one Teacher
    - Every Mother has at least one Child
* Cardinality Constraints
  + Express the number of entities to which another entity can   
    be associated via a relationship set.
  + Cardinality Constraints - the number of instances of one entity that can or must be associated with each instance of another entity.
    - Minimum Cardinality
    - If zero, then optional
  + If one or more, then mandatory
    - Maximum Cardinality
    - The maximum number

Chart, diagram

Description automatically generatedTable

Description automatically generated

Normalization:

* Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that avoid unnecessary duplication of data
* The process of decomposing relations with anomalies to produce smaller, well-structured relations
* Anomaly: problem in un-normalized databases
* A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
* Goal is to avoid anomalies
* General rule of thumb: A table should not pertain to more than one entity type.
* Insertion Anomaly: adding new rows forces user to create duplicate data
* Deletion Anomaly: deleting rows may cause a loss of data that would be needed for other future rows
* Update Anomaly: changing data in a row forces changes to other rows because of duplication
* The main function of a database is to STORE and RETRIEVE information. We need to be able to do this efficiently and accurately.
* A database that is not well formed is prone to the following issues   
  due to anomalies:
  + Efficiency - increased query time
  + Data integrity – missing, inaccurate, and duplicate information
  + Ease of use - increased effort in writing queries to store and   
    retrieve date
* Functional dependency: The value of one attribute (the determinant) determines the value of another attribute
* Candidate key: A unique identifier. One of the candidate keys will become the primary key
  + E.g., perhaps there is both credit card number and SIN# in a table, in this case both are candidate keys.
* First normal form:
  + No multivalued attributes
  + Every attribute value is atomic
  + No multivalued attributes
  + Recall multivalued attributes are lists
  + Storing a list in a database means duplicate data and empty/null entries.
* Second normal form:
  + 1NF PLUS every non-key attribute is fully functionally   
    dependent on the ENTIRE primary key
    - Every non-key attribute must be defined by the entire key, not   
      by only part of the key
    - No partial functional dependencies
* Third normal form:
  + 2NF PLUS no transitive dependencies (functional dependencies on non-primary-key attributes)
  + Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
  + Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as a foreign key in the old table
* Boyce-Codd Normal Form (BCNF):
  + Relation has more than one candidate key, anomalies may result even though that relation is in 3NF.
  + A relation is in Boyce-Codd normal form (BCNF) if and only if every determinant in the relation is a candidate key.
  + Simply put, a relation is in BCNF when every attribute/field depends on the key and nothing but the key.

Diagram

Description automatically generated

Database design:

* Relationships:
  + One to One
    - No resolving necessary
  + One to Many
    - Optional to resolve to one to one in rare cases, not common and likely undesirable due to information loss
  + Many to Many
    - Resolving is almost always desirable
* Design process:
  1. Identification
  2. Describe
  3. Relationship
  4. Normalization
  5. Review
* Datatypes:
  + Char/character – fixed length string, always occupies defined space
  + Varchar/character varying – variable length string, occupies string length + 1
  + Text – used to large chunks of text
    - Number types
    - Integers – whole numbers. INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT
    - Decimal/numeric
    - Float/double
    - Bit
  + Date and Datetime
  + Timestamp – special version of datetime that respects time zones
  + Time and Year
* Composite key. A key that is composed of two or more attributes.
* Natural key. A key that is formed of attributes that already exist in the real world.
* Synthetic key (A.K.A. Surrogate key). A key with no business meaning.
* Primary key. The preferred key for an entity type.
* Foreign key. One or more attributes in an entity type that represents a key, either primary or   
  secondary, in another entity type.

Indexes:

* Most queries only require a small amount of information from a database
* What if the only way to get the information was to search the ENTIRE database – INEFFICIENT!
* Indexes help us speed up our queries so we don’t have to search the entire database
* Analogy: Instead of searching an entire textbook for the page we’re looking for, we have an index at the back of the book that allows us to look up the topic and page number.
* Definition: A data structure that is used to speed up data retrieval. It typically contains a list of keys used to identify columns in tables.
* Index: a table or other data structure used to determine in a file the location of records that satisfy some condition
* Primary keys are automatically indexed
* Other fields or combinations of fields can also be indexed; these are called secondary keys (or nonunique keys)
* Uses Tree search
* Unique (primary) Index: Typically done for primary keys, but could also apply to other unique fields
* Nonunique (secondary) index: Done for fields that are often used to group individual entities (e.g. zip code, product category)
* Rules for Using Indexes  
  1. Use on larger tables  
  2. Index the primary key of each table  
  3. Index search fields (fields frequently in WHERE clause)  
  4. Fields in SQL ORDER BY and GROUP BY commands  
  5. When there are >100 values but not when there are <30 values

6. Avoid use of indexes for fields with long values; perhaps   
compress values first  
7. If key to index is used to determine location of record, use   
surrogate (like sequence number) to allow even spread in   
storage area  
8. DBMS may have limit on number of indexes per table and   
number of bytes per indexed field(s)  
9. Be careful of indexing attributes with null values; many DBMSs   
will not recognize null values in an index search

Graphical user interface

Description automatically generated with medium confidence

Views:

* Views are relations, except that they are not physically stored.
* For presenting different information to different users
* A view is not a true table, it does not contain data
* When we query a view, it accesses the source table to capture the data
* This means that if data needs to be modified, it must be modified at the table level
* Types of Views
  + Virtual views:
    - Used in databases
    - Computed only on-demand – slower at runtime
    - Always up to date
  + Materialized views
    - Used in data warehouses
    - Precomputed offline – faster at runtime
    - May have stale data



Text

Description automatically generated with medium confidence

Text

Description automatically generated